# METADATA MODEL FOR DATABASE QUERY ON DYNAMIC DATABASE FEDERATION IN DATA GRID ENVIRONMENT BASED ON ONTOLOGY FRAMEWORK : A TEXT SIMILARITY APPROACH

Krisna Adiyarta[1], Naomi Salim[2]

[1,2]Faculty of Computer Science and Information Systems
University Teknologi Malaysia
81300 Skudai, Johor

[1]adiyarta@yahoo.com, [2]naomie@utm.my

**Abstract** : One of the aims of the Grid is to promote the open publication of scientific data. If this is realized then it is expected that many of the advances to flow from the Grid will be



requirements of early Grid projects has concluded that the prospect exists for millions of data resources and petabytes of data being accessible in a Grid environment [22]. Support for federating data resources is therefore vital to the success of the Grid. Grid database federation requires a highly dynamic federation. It means that, Grid database should accept changes occur in a federated database. Therefore, database systems need metadata management that has a capability to organize and maintain stored information in term of a dynamic federation. In this paper we discuss our proposed of metadata management model that provide a dynamic framework for grid database federation. We present a model of metadata management in term of an ontology based data integration framework.

**Keyword** : Dynamic Database Federation, Grid Database, Metadata, Ontology, Similarity Measurement

## 1. INTRODUCTION

Grid Technology has found uses in a wide area of applications that usually address large scale, process and data intensive problems. Grid computing makes science are made possible largely through the collaborative efforts of many researchers in a particular domain. It means that the collaborations of hundreds of scientists in areas coming together and sharing a variety of resources within a collaboration in pursuit of common goals. These resources are distributed and can encompass people, scientific instruments, compute and network resources, applications, and data.

Metadata is information that describes data [5]. It is essential for facilitating data management tasks that are involved in publishing and discovering data. A metadata management system is used to maintain the metadata and provide interfaces for other system components to access the metadata. Metadata is widely used by various systems components to provide wide-ranged support for process automation, user-machine interaction and decision support. Data sources in a grid environment can be heterogeneous in syntax, schema, or semantics, thus making data interoperation a difficult task. This circumstances, shows that data Grid requires more sophisticated metadata systems and tools, such as adding semantic to metadata which assist software agent to discover and access the sources. The result is likely to be a Semantic Grid [25] that is analogous to the Semantic Web [6] in which web content can be expressed in a form that can be understood, interpreted and used by software agent.

As to Grid Based Federated Database systems, one major goal of metadata management system is to handle interoperability between different grid members. In traditional data integration, the solution proposed by the database community aims to identify and treat the relationships between schemas (schemas mapping). Therefore, mappings should be generated between pairs of conceptual schemas. However, this solution became impracticable due to the necessary amount of different mappings when working with a great number of databases. The common solution is based on the use of a global schema containing unification elements that correspond to each original database conceptual schema. Another approach for the problem of semantic interoperability is based on the use of ontologies to expose the implicit knowledge of applications [19],[13].

The paper is organized as follows: in section 2 we will discuss the overview of Grid Database, in term of concept, integration and metadata. In section 3, we discuss role of ontology in a data integration point of view.. As the main discussion of this paper, in section 4, we discuss our metadata managemen model which could be employed in a dynamic data federation. Finally, conclusion and further research is described in section 5.

## 2. DATABASE ON GRID

Grid-Database concept, which "is a collection of one or more databases logically interrelated (distributed over a grid environment) which can also be heterogeneous and accessible through a Grid-DBMS front end. It represents an extension and a virtualization of the Database concept in a grid environment". At the highest level the Grid-DBMS consists of two components, one mainly hardware and another one specifically software. First, the Legacy

System: it brings together both hardware elements (pc, workstations, storage, network, etc.) of the grid infrastructure and legacy software for data management (i.e. DBMSs already installed). Second, the Grid-DBMS middleware, grid middleware which allows managing, accessing, integrating, optimizing and reconfiguring data sources installed within a Legacy System [31].

### 2.1 Database Federation

The Grid offers new opportunities and raises new challenges in data management that arise from the large scale, dynamic, autonomous, and distributed nature of data sources. A Grid can include related data resources maintained in different syntaxes, managed by different software systems, and accessible through different protocols and interfaces. Due to this diversity in data resources, one of the most demanding issue in managing data on Grids is reconciliation of data heterogeneity [3]. Therefore, in order to provide facilities for addressing requests over multiple heterogeneous data sources, it is necessary to provide data integration models and mechanisms.

The goal of a data integration system is to combine heterogeneous data residing at different sites by providing a unified view of this data. The two main approaches to data integration are federated database management systems (FDBMSs) and traditional mediator/wrapper-based integration systems. A federated database management system (FDBMS) [28] is a collection of cooperating but autonomous component database systems (DBSs). The DBMS of a component DBS, or component DBMS, can be a centralized or distributed DBMS or another FDBMS. The component DBMSs can differ in different aspects such as data models, query languages, and transaction management capabilities. Traditional data integration systems [21] are characterized by an architecture based on one or more mediated schemas and a set of sources. The sources contain the real data, while every mediated schema provides a reconciled, integrated, and virtual view of the underlying sources. Moreover, the system includes a set of source descriptions that provide semantic mappings between the relations in the source schemas and the relations in the mediated schemas [24].

Data integration on Grids presents a twofold characterization:

1. data integration is a key issue for exploiting the availability of large, heterogeneous, distributed and highly dynamic data volumes on Grids;
2. integration formalisms can benefit from a Grid infrastructure, such as n OGSA-based, since it facilitates dynamic discovery, allocation, access, and use of both data sources and

computational resources, as required to support computationally demanding database operations such as query reformulation, compilation and evaluation.

Data integration on Grids has to deal with unpredictable, highly dynamic data volumes provided by unpredictable membership of nodes that happen to be participating at any given time. So, traditional approaches to data integration, such as FDBMS [28] and the use of mediator/wrapper middleware [32], are not suitable in Grid settings. The federation approach is a rather rigid configuration where resources allocation is static and optimization cannot take advantage of evolving circumstances in the execution environment. The design of mediator/wrapper integration systems must be done globally and the coordination of mediators has been done by a central administrator which is an obstacle to the exploitation of evolving characteristics of dynamic environments. As a consequence, data sources cannot change often and significantly, otherwise they may violate the mappings to the mediated schema.

### 2.2. Integration Database Into The Grid

The nature of grid computing shows the creation of a diversity of sites and data formats. Virtualization introduces transparencies that should be considered when designing any integration [2]. The integration framework on grid is an implementation of a service based framework [30]. The framework performed by supporting the creation of virtual databases that present the same service interface to applications as do the individual, unfederated, database systems. One advantage of a service-based framework is however that each DBS made available on the Grid can provide a metadata service that gives information on the range of services and operations that it supports.

In the integration, a "virtual database system" acts as the interface for database on grid. Figure 1 show the "virtual database system" that integrates database system on grid. The "virtual database system" defined by him is actually is a service-based approach. It extends the basic service in term of federated database. Virtual database system has exactly the same service interface as the database system but does not actually store any data. Instead, calls made to the virtual database system services are handled by service federation middleware that interacts with the service interfaces of the individual database system that are being federated, in order to compute the result of the service call. Because the virtual database system has an identical service interface to the "real" database system, then it is possible for a virtual database system to federate the services of both "real" database system, and other virtual database system.

Basically, framework discussed above is a service-based. Two primary services correspond to our issue are metadata and query service. This service provides access to technical metadata about the DBS and the set of services that it offers to Grid applications. Query Service supports the compilation and evaluation of queries that combine data obtained from several query services on the Grid and local data. Hence, the use of metadata to locate data has important implications for integrating databases into the Grid because it promotes a two-step access to data. In step one, a search of metadata catalogues is used to locate the databases containing the data required by the application [30]. That data is then accessed in the second step. A consequence of two-step access is that the application writer does not know the specific database system that will be accessed in the second step. Therefore the application must be general enough to connect and interface to any of the possible database system returned in step one.

### 2.3 Metadata Management In General

One of the challenges of the shared environments is to identify and locate the subset of data objects that is of interest to a particular data analysis activity. The standard solution to this problem is to describe the characteristics of each data object with one or more attributes, or "metadata," and to use this metadata as the means of identifying relevant data objects. Metadata allows collaborations to publish data with enough information to be able to identify the desired data products.

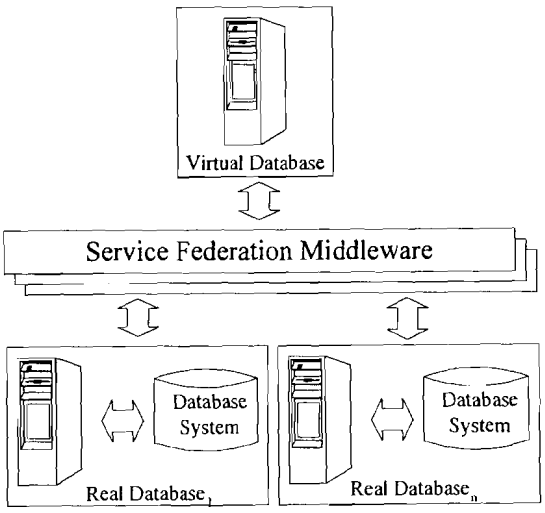


Figure 1: Database Integration model on the Grid

### 2.3.1 Category of Metadata.

Metadata, the term for describing 'data about data' is essential to meeting many of the Grid requirements [3]. It is essential for facilitating data management tasks that are involved in maintaining the integrity and consistency of data, and for tasks involved in publishing and discovering data. Metadata is referenced when performing processing, retrieval, analysis and interpretation of data, and it is important for establishing the ownership, currency, validity, and quality of data. Metadata is also essential to the development of Grid services because it enables data operations to be abstracted to a sufficient degree that services can be created and made reusable. This facility makes it possible to access and manipulate data content without knowing where it is physically located, or how it is structured.

Several types of Metadata are important for the development of Grid data access and integration services:

1. Technical metadata defines the location of data sources and resources; the physical data structure, organization and grouping of data items into logical records; and those characteristics of the data that are important in deciding how data is best accessed and transported. Technical metadata also defines data currency and history; i.e. versions, and ownership of data.
2. Contextual metadata defines naming conventions, terminologies and ontologies through which data can be logically referenced. Contextual metadata increases the quality and reliability of data because the definitions conform to agreed syntax and semantics, and also record structural associations and relationships within the data between definitions, and to define rules for conflicts between mappings.
3. Derived metadata defines the context and meaning of data derived from any other data. This type of metadata is commonly used in data warehousing environments, when it is often more efficient to store derived data than to recalculate the values dynamically each time they are required.
4. Mapping metadata defines equivalences between discrete contextual metadata definitions, and between contextual and technical metadata.

### 2.4. Metadata Management In Grid Database Federation

Metadata Management has been widely discussed in different areas, such as data warehouse, digital library, educations, scientific and business oriented area. Support for federating data resources is therefore vital to the success of the Grid. Several papers has described that the

alternative of forcing each application to interface directly to a set of databases and resolve federation problems internally would lead to application complexity, and duplication of effort [16],[4],[11]. Three major aspects considered for a metadata management are data heterogeneity, grid as service oriented architecture and semantic grid.

### 2.4.1 Heterogeneity

As to Grid Based Federated Database systems, one major goal of metadata management system is to handle the heterogeneity and allow interoperability between different grid members by providing the degree of transparency to users who access the information. Heterogeneity exists at different levels in a distributed system. In [20], the author identifies four different types of heterogeneity in the case of data sources within digital libraries. These types heterogeneity applies to all kinds of grid systems.

1. System heterogeneity arises from different hardware platforms and operating systems.
2. Syntactic heterogeneity arises from the presence of different protocols and encodings used with the system.
3. Structural heterogeneity originates from the data organized according to different models and schemas.
4. Semantic heterogeneity originates from different meanings given to the same data, especially because of the use of different metadata schemas for categorizing the data.

### 2.4.2 Service Oriented Architecture

As [16[ and [14] stated, Service Oriented Architecture is considered an important solution for interpretability and to solve the problem of the high degree of heterogeneity. In the past several years, Grid technology is under rapidly progressing of maturation and standardization around the banner of the Open Grid Services Architecture. OGSA builds upon standard web services and extending its power by introducing transient, stateful service instances (Grid Service). It allows maximum interoperability between different system components. Within OGSA, including metadata management, everything is represented as a Grid Service.

A service might contain metadata that describes its capabilities, interfaces, provenance, performance, security and access policies, and the metadata could be distributed at all levels of the Grid [3]. A service could also publish its metadata to metadata repositories that can then be queried to discover resources and services that have specified characteristics. As to

the task of metadata services, the author claims that metadata services are divided in registry, information aggregation, metadata (Semantic Grid), curation and provenance areas.

### 2.4.3 Semantic Grid

Discussion on Grid [15],[14],[16] emphasis the importance of introducing knowledge service into metadata management in order to give machines the capability to make well-informed decisions about data and processes based on logical inferences. Semantic Grid is a term used to describe a metadata-rich environment for grid computing. The integration of grid and semantic web meta-data and ontology technologies has been considered as a major trend in metadata management.

## 3. ONTOLOGY BASED DATABASE INTEGRATION SYSTEM

Data integration provides the ability to manipulate data transparently across multiple data sources. Ontologies were developed by the Artificial Intelligence community to facilitate knowledge sharing and reuse [9],[18]. Carrying semantics for particular domains, ontologies are largely used for representing domain knowledge. A common use of ontologies is data standardization and conceptualization via a formal machine-understandable ontology language. This section we will discuss the a general framework of data integration approach using ontology.

### 3.1. Ontology and Metadata

In a philosophical sense Ontology is the discipline that concerns the definition of a particular system of categories accounting for a certain vision of the world [7]. Under this definition, an ontology is independent of a language used to describe it. The artificial intelligence community, in contrast, defines an ontology in regard to a specific vocabulary that describe a certain reality. An ontology defined as an explicit specification of a conceptualization [17]. Distinguishing a conceptualization (**C**) from an ontology, ontology can be defined and described as "a logical theory designed to account for the intended meaning of a vocabulary; i.e., its ontological commitment (**K**) to a particular conceptualization of the world." They suggested that a conceptualization is "an intensional semantic structure which encodes the implicit rules constraining the structure of a piece of reality." Figure 2 clarifies the relationship among conceptualization, language (**L**), and ontology [18]. A relationship

between the philosophical and engineering senses of an ontology exists if a conceptualization is associated with the philosophical sense of an ontology.

In both computer science and information science, an ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts. It is used to reason about the objects within that domain. Ontologies generally describe: Things or individuals, classes, properties, attributes, relations or interaction [7].

When a database is published some of the metadata will be installed into a catalogue (or catalogues) that can be searched by applications looking for relevant data. The use of metadata to locate data has important implications for integrating databases into the Grid. Integration of metadata is required since it is represented in several form to achieve the quality of information. Integration of metadata means that metadata should be described in different perspective.

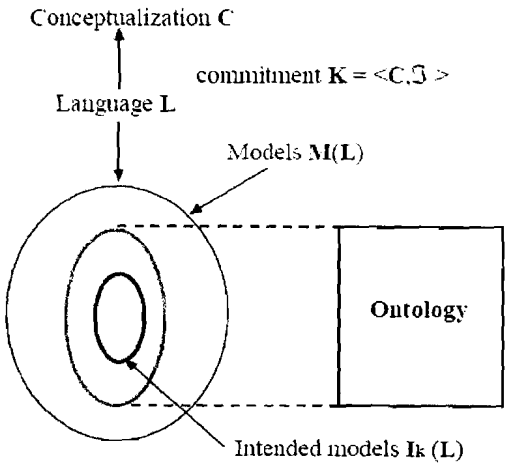


Figure 2: Relationship among conceptualization, language, and ontology [18]

In Grid service point of view, there are three perspectives of metadata are conceptual, logical, and physical [23]. Figure 3 illustrates the perspectives of metadata. The conceptual perspective represents the semantical models of the information grid, i.e. the ontologies. The logical perspective models the structure of the data of the individual resources of the information grid. The logical schema will be defined as view on the conceptual schema. The horizontal link "isViewOn" between Element and Class represents this. The physical perspective describes where the data is located and how it can be accessed. This is done by specifying the type of service of the resource. The important information is represented by the

horizontal link to the logical perspective, which provides the information what kind of schema is used in the result, and which arguments are required for a method provided by the service.

Conceptual perspective is intended to capture knowledge about a real-world domain. Hence, ontology is used. The term ontology is sometimes used to refer to a body of knowledge describing some domain, typically a commonsense knowledge domain, using a representation vocabulary. In other words, the representation vocabulary provides a set of terms with which to describe the facts in some domain, while the body of knowledge using that vocabulary is a collection of facts about a domain [9].

### 3.2. Ontology Representation

Generally, ontology is a model for describing the world that consists of a set of types, properties, and relationship types. Therefore, ontology is related to the conceptual data modeling. The aim of conceptual model is to express the meaning of terms and concepts used by domain experts to discuss the problem, and to find the correct relationships between different concepts. This is also called semantic model. The conceptual model attempts to clarify the meaning of various usually ambiguous terms, and ensure that problems with different interpretations of the terms and concepts cannot occur. Such differing interpretations could easily cause the software projects that are based on the interpretation of the concepts to fail. Once the domain concepts have been modelled, the model becomes a stable basis for subsequent development of applications in the domain.

The conceptual model could be described with a class diagram. A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. In a class diagram, classes are represented by boxes with three parts: the name of the class, the attributes of the class (specified by their name, type and visibility). For the purposes of representing ontologies, all attributes can be considered to have public visibility, an ontology is a shared public view of a domain. There are two types of relationship that may be used between classes

- generalization, represented by lines with large hollow arrow heads pointing to the super class;
- association, represented by solid lines between two classes with optionally named ends, or roles;

### 3.3. Query Language Model

The semantics of a class diagram, as an ontology representation, defines the constraint specification of a database. Formally, a database is defined over a fixed (infinite) alphabet of symbols, each one denoting a semantic value. The database assigns to each class a subset of the alphabet, to each attribute of a class a binary relation over the alphabet. The set of objects assigned by the database to a class, attribute is called the set of its instances in the database.

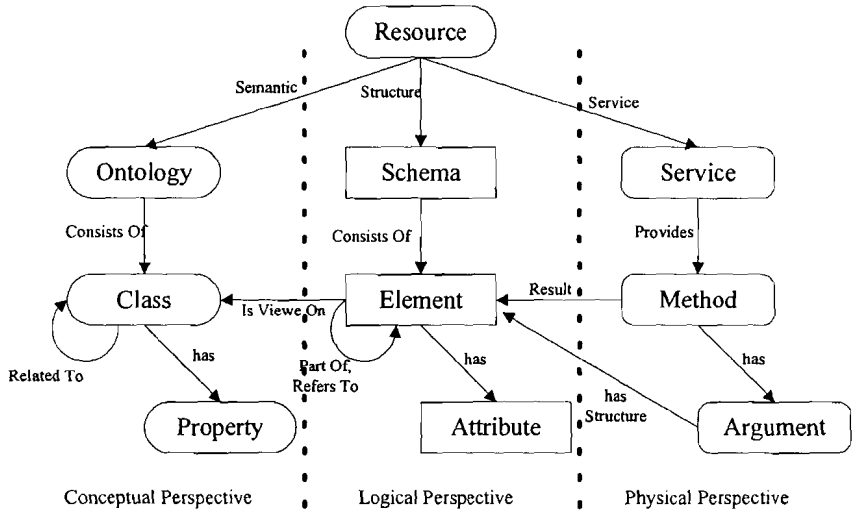


Figure 3: Perspectives of Metadata [23]

Conjunctive query statement could be used as the query statement model. Conjunctive queries in relational database systems are the queries of first-order logic that are built from atomic formulas by means of conjunctions and existential quantification only [29]. Thus, the generic conjunctive query takes the form

$$h :- p_1, \ldots, p_n$$

where $h$ is a literal whose arguments are distinct variables, $p_1, \ldots, p_n$, are literals, $h$ is the head, and $p_1, \ldots, p_n$ is the body. Variables in the head are distinguished.

Queries posed to a grid database D are expressed in terms of a query language $Q$ over the alphabet $A$ and are intended to extract a set of tuples of elements of the interpretation $\Delta$ obtained from the catalog. Thus, every query has an associated arity, and the semantics of a query $q$ of arity $n$ is defined as follows. The answer $q^D$ of $q$ to $D$ is the set of tuples

$$q^D = \{(c_1, \ldots, cn) \mid \text{for all I} \in sem(\text{D}), (c_1, \ldots, c_n) \in q^I\}$$

where $q^I$ denotes the result of evaluating $q$ in the interpretation $I$.

Conjunctive query could be used as the query language model. Conjunctive queries in relational database systems are the queries of first-order logic that are built from atomic formulas by means of conjunctions and existential quantification only. Thus, the generic conjunctive query takes the form

$$h :- p_1, \ldots, p_n$$

where $h$ is a literal whose arguments are distinct variables, $p_1, \ldots, p_n$, are literals, $h$ is the head, and $p_1, \ldots, p_n$ is the body. Variables in the head are distinguished.

Query on a data integration system is expressed in a query language $Q$ over the alphabet $A$ is based in the global ontology $G$, such as that presented in the class diagram. Queries posed to the ontology integration system are expressible as project-select-join queries where the selection conditions are restricted to equality. Predicates are divided into Extensional Database (EDB) predicate or stored relation and Intentional Database (IDB) predicate whose relation is constructed by rules. An atomic formula whose arguments are variables or constants is a (positive) literal. For an ontological language model, the predicates in the atoms are the so-called concepts of the conceptual schema, its classes and attributes, could be presented as :

- Each class $E$ in $G$ has an associated predicate $E$ of arity 1. Intuitively, $E(c)$ asserts that $c$ is an instance of class $E$.
- Each attribute $A$ for an class $E$ has an associated predicate $A$ of arity 2. Intuitively, $A(c, d)$ asserts that $c$ is an instance of class $E$ and $d$ is the value of attribute $A$ associated to $c$.

Further discussion of these IDB structure will describe on section 3.5.

For an example, consider the class diagram shown in Figure 4, depicted in the usual graphical notation. Suppose we want to know the names of the employees who live in "Kuala Lumpur". The corresponding conjunctive query which is evaluated for schema 1 in figure 4 is

$q(y) \leftarrow \text{Employee}(x), \text{pname}(x, y), \text{Lives In}(x, z), \text{cname}(z, \text{"Kuala Lumpur"})$

Example 1: Query

Conjunctive queries can be viewed as a generalization of tableaux in which the evaluation could be performed. Tableaux are similar to conjunctive queries, and every relational expression over the operators select, project, and join can be represented by a tableau. Please refer to [27] and [1] for more detail according to these discussion. Figure 5 shows the plain relation retrieve as the extension of query expressed in example 1.

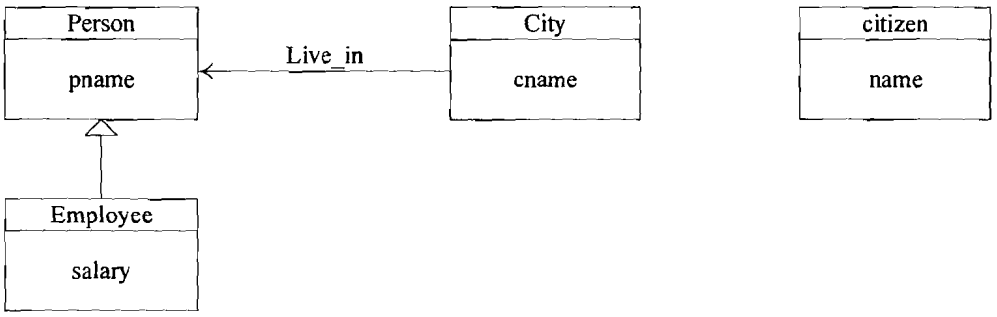


Figure 4: Class Diagram

### 3.4. Query Answering Process

Previously, we have discussed that one of the most important aspects in the design of a data integration system is the specification of the correspondence between the data at the sources and the global schema. In some sense, the mapping explicitly tells the system how to retrieve the data when one wants to evaluate the various elements of the conceptual schema. It assists the user to use directly the global schema, unaware of the separate databases behind the view. Unlike ordinary tables (base tables) in a relational database, a view is not part of the physical schema: it is a dynamic, virtual table computed or collated from data in the database. Changing the data in a table alters the data shown in the view.

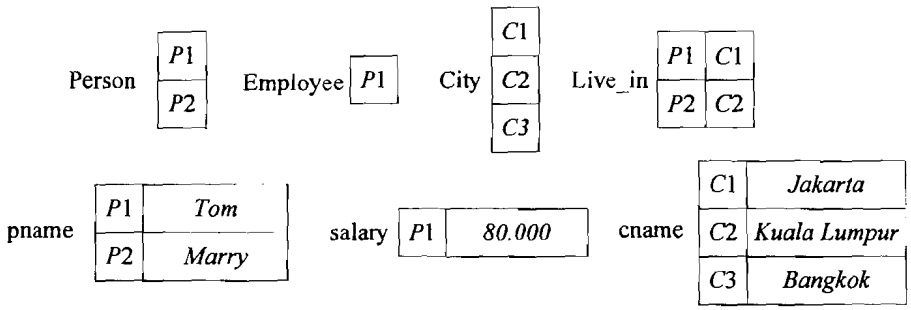


Figure 5: Extension of relations of Schema 1 on figure 4

Data integration systems could adopt a simpler data model, such as a plain relational data model for expressing the global schema. In this case, the data retrieved from the sources trivially fits into the schema, and can be directly considered as the unique database to be processed during query answering. The canonical database is a special database that represents all possible global databases legal for the data integration system. Therefore, Discussion on data integration has been showed a strategy for computing the certain answers to a query based on the canonical database [8],[10].

Figure 6 illustrates the query answering process. Let $q$ be a conjunctive query posed to data integration system $I$ and let $D$ be a source database for $I$. At the beginning, parsing and decomposing the query is performed to obtain predicates $p$ of the query $q$. Based on all $p$ the expansion view $V$ is obtained using mapping definitions for each predicate. Operation on source database $D$ is performed by evaluating or unfolding these expansion, i.e. data service execution, view $V$ over the source database $D$ using their local ontologies. The various relations are obtained form these view evaluation over source, called as the retrieved global database ret($I$,$D$). A canonical database is established to represents all possible global databases legal for the data integration system from all ret($I$,$D$). Finally the query $q$ is posed to canonical database to be evaluated to obtain the result.

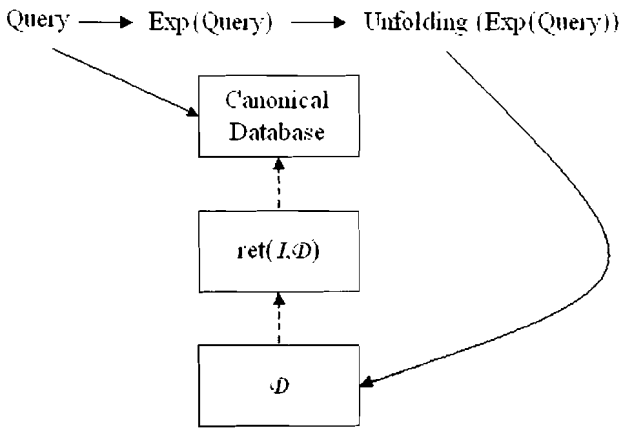


Figure 6: Query Answering Process

### 3.5. Accessing Data Using Conceptual Model

Database on Grid facilitate the sharing of data resources to enable data collaboration. Basically, it presents a form of a data integration model to access the distributed data resource. We can adopt a global-centric approach to perform data integration systems. In such systems, sources are databases, the global ontology is actually a database schema, and the mapping is specified by associating to each database component in the global schema one relational query over the source database. This mechanism allow for a simple query processing strategy, which basically reduces to unfolding the query using the definition specified in the mapping, so as to translate the query in terms of accesses to the sources.

The data integration systems are characterized by an architecture based on a global schema and a set of sources. The sources contain the real data, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. Global-centric approach is common approach used in data integration system. From the modeling point of view, the global-centric approach is based on the idea that the content of each element $g$ of the global ontology is characterized in terms of a view $q$ over the sources. The mapping explicitly tells the system how to retrieve the data when one wants to evaluate the various elements of the conceptual schema. Therefore, if we have a query language $V$ over the alphabet $A$, and the mapping between the conceptual and the source ontologies is given by associating to each term in the conceptual model (ontology) with a view over the source ontologies. The intended meaning of associating to a term $C$ in conceptual schema $G$ a query $V$ over database, is that such a query represents the best way to characterize the instances of C using the concepts in database.

In term of specifying the semantics of ontology for a database system (ODBS), we start with a model of the source ontologies, and the crucial point is to specify which the models of the conceptual model ontology. Thus, for assigning semantics to an ontology for a database system ODBS $O = <G, S, M>$ we start by considering a database with model $D$ for $O$, in other words, an interpretation that is a model for all the theories of database. We call global interpretation for $O$ any interpretation for $G$. The semantics of $O$, denoted $sem(O)$, is defined as follows:

$$sem(O) = \{ I \mid \text{there exists a model } D \text{ for } O \text{ s.t. I is a model for } O \text{ wrt } D \}$$

Mapping definition presents a single unified view to access data sources. A view could be seen as a virtual or logical table composed of the result set of a query. The definition of mapping $M$ between global ontology $G$ and source ontology $S$ is given by a set of correspondences in the form of $<C; V>$, where $C$ is a concept in the global ontology and $V$ is a query over source $S$. More precisely,

- The mapping associates a query of arity 1 to each class $E$ of $G$. The query retrieves ($x$) from the extension of the local ontologies. The main purpose of an this IDB is to list all the surrogates of entities that have $E$ type and are currently recorded in the database. The value of $x$ is a surrogate or identity. It is property of an object which distinguishes it from all others. A surrogate is a unique value assigned to each entity. If two views retrieve the same surrogate value then they represent the same entity in the modelled universe.
- The mapping associates a query of arity 2 to each entity attribute $A$ of $G$. The query retrieves the pair ($x$; $y$) from the extension of the local ontologies, this means that $y$ is a value of the attribute $A$ of the class instance $x$. Thus, the first argument of the query corresponds to the instances of the class for which $A$ is defined, and the second argument corresponds to the values of the attribute $A$. The single values of $A$ represent a value of object properties/relationships (binary predicates relating individuals) or datatype properties/attributes (binary predicates relating individuals with values such as integers and strings).

Consider the conceptual schema shown in Figure 4, example 2 shows a mapping definition of global schema G of a data integration system $I = <G, S, M>$ where S are constituted by

- s1, s2, s3, s4, s5, s6, s7, s8, s9 for real local database access using schema 1

- citizen, name are views over another database with schema 2 to access their data source.

The mapping *M* is as follows:

$$
\begin{aligned}
\text{person (a)} &\leftarrow s1(a) \vee \text{citizen(a)} \\
\text{pname(a,b)} &\leftarrow s1(a), s2(a,b) \vee \text{name(a,b)} \\
\text{city(c)} &\leftarrow s3(c) \\
\text{cname(c,d)} &\leftarrow s3(c), s4(c,d) \vee s1(c), s5(c,d) \\
\text{live\_in(a,c)} &\leftarrow s1(a), s6(a,f), s7(f,c) \\
\text{employee(g)} &\leftarrow s8(g) \\
\text{salary(g,h)} &\quad\; s8(g), s9(g,h)
\end{aligned}
$$

Example 2: Mapping Definition

## 4. METADATA MANAGEMENT FOR DYNAMIC DATABASE FEDERATION

Grid database federation is different from a conventional database federation. Grid database federation requires a highly dynamic federation. Changes occur in a federated database in grid environment when attributes are added, removed or modified at local database level or when an entire database is added or removed. Database systems need metadata to define the stored information and services. In federated database systems, metadata generally means the virtual schemas which reside above the local schema in the federated architecture. In this section we discuss our use of metadata to provide a dynamic architecture for federated database systems.

### 4.1. Metadata Model

We have discussed, that our database integration is presented as a single integrated view with a single federated schema (global schema). Therefore, the user would then use directly, unaware of the separate databases behind the view. Unlike ordinary tables (base tables) in a relational database, a view is not part of the physical schema: it is a dynamic, virtual table computed or collated from data in the database. Changing the data in a table alters the data shown in the view. Just like functions (in programming) provide abstraction, views can be used to create abstraction. Also, just like functions, views can be nested, thus one view can aggregate data from other views.

Dynamic federation on the Grid means that the many applications could dynamically select a set of databases to access based on the results of queries of metadata catalogues. This, and the fact that the set may be very large, places a premium on fully automatic federation without user input. This needs resolution at the interface level that the federation middleware communicate with the different interfaces offered by the databases to be federated.

The use of metadata to locate data has important implications for databases access and integration because it promotes a two-step access to data. In step one, a search of metadata catalogues is used to locate the databases containing the data required by the application. That data is then accessed in the second step. A consequence of two-step access is that the application writer does not know the specific database system that will be accessed in the second step. Therefore the application must be general enough to connect and interface to any of the possible database system returned in step one.

Typically, class diagrams are accompanied by comments or notes about each class, and/or its relationships, attributes. To enable adding information to a model that otherwise cannot be represented in conceptual model, in some modeling language, such as UML, provides a notes capability. A note can contain any type of information. The note is typically attached to some element in the diagram. Generally these notes is stored as text data as the description of an element that maps to the elements in the diagram, such as in IBM Rational. In our approach we employ these note to express the intended semantics of the concept.

Intuitively, in specifying the map of contextual metadata to physical data structures and schemas, discussed in section 2.3.1, we start with a model of ontologies, and the crucial point is to specify which the models of the physical and data structure or conceptual model. Thus, for assigning semantics to an ontology for contextual metadata of an ontology

$$MD = <C, V, N>$$

we start by considering the model $D$ for $MD$, in other words, an interpretation that is the model for all the theories of the database. The mapping between concept $C$ and physical data structures or schemas is given by the correspondence between concept $C$; and its view $V$, where $C$ is a concept in the ontology and $V$ is a query over physical data structures and schemas. Next, $N$ ,a notes, presents the description in natural language to express the intended semantics of the concept $C$ . In term of two steps access, these comments will be evaluated linguistically to determine the similarity between concepts among database being federated.

### 4.2. Similarity Measures of Concept Descriptions

Previously, we have presented our metadata as a tuple of $<C, V, N>$ where $C$ is a concept, $V$ is a query to express the extension concept $C$ and $N$ is a short text that presents the intended semantics of the concept $C$. The primary task of our proposed metadata management for dynamic federation is the mechanism for users to access data content using ontological references. Hence, we use $N$ as the metadata in order to specify a matching process between ontologies when performing integrated or federated queries against multiple data resources. These matching mechanism is performed by a measuring the similarity between $N$. For this reason, the text similarity measures must be considered as approach for data integration or data federation in different classifications and ontologies. There are two aspects pertain to the approach: a) Representation models. b) Similarity measures.

#### 4.2.1. Text Representation

Generally in Information Retrieval, conventional text representation models focus on whether a document contains specific keywords, and their appearance frequencies. We could adopt The vector space model for our metadata management model. The vector space model [26], documents are represented by vectors containing the frequency of all possible words (features) in a document set. A document is represented as a vector. Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. One of the best known schemes is tf-idf weighting.

In the classic vector space model, the term specific weights in the document vectors are products of local and global parameters. The model is known as term frequency-inverse document frequency model. The weight vector for document $d$ is

$Vd = [w_{1,d}, w_{2,d}, \ldots w_{N,d}]^T$ , where

$$w_{t,d} = tf_t \cdot \log \frac{|D|}{|\{t \in d\}|}$$

and

- $tf_t$ is term frequency of term t in document $d$ (a local parameter)

- $\log \frac{|D|}{|\{t \in d\}|}$ is is inverse document frequency (a global parameter).
- $|D|$ is the total number of documents
- $|\{t \in d\}|$ is the number of documents containing the term $t$.

### 4.2.2. Similarity Measures

Relevancy rankings of the metadata notes, as a short text documents, in a keyword search can be calculated, using the assumptions of document similarities theory, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as same kind of vector as the documents. Hence, similarity measures are used to determine distances between documents, after transforming the textual data into a useable and intelligible format. In vector space model, the feature space constitutes a geometric space where documents are represented as points in a multidimensional space. Thus, measuring the similarity can be easily calculated. A measure are often used is the cosine measure [12]. The cosine measure is defined as

$$\cos\theta = \frac{x \bullet y}{|x|\,|y|}$$

where (x.y) denotes the vector dot product of x and y, and |x| and |y| are the lengths of vectors x, y, respectively.

## 5. CONCLUSION AND FUTURE WORK

In this paper we have discussed our framework that is employed to a query service for a grid database. We have discussed that the basic elements in the metadata management framework are the conceptual schemas and its mapping to the data sources. The schema, it is presented using conceptual model that provides a reconciled, integrated, and virtual view of the underlying sources. Mapping between global schema and data source is given by a set of correspondences between concept in the global conceptual model and a query over sources (local or another grid database).

In order to access the data, the expression of query language for database use the alphabet based on the global ontology. It is expressed in the form of atoms that refer to the concepts on global schema. Query processing presents the algorithm for reasoning about both the query

and the global ontology in order to infer which tuples satisfy the query in all models. Primary task in our metadata management model is on the element matching approach in order to obtain view definition to access the database. We apply text similarity measures that take as input of two description between elements from conceptual global schema and conceptual source schema and performs matching on every descriptions.

The goal of our work is to introduce, build, and demonstrate a novel metadata management for grid database federation, particularly for database on grid environment that is based on semantic understanding of these description contents of concept descriptions. The system is composed of components that facilitate semantic analysis of text and measuring similarity between documents. Future work aims is to show the working of the method and how this system can provide better performance. These results depend crucially on the choice of effective term weighting systems. Currently, we are studying the problem of measuring the similarity between short segments of text. We looked at various types of text representations and the similarity measures based on these representations.

## REFFERENCES


[1] Aho *et al* (1979), "Efficient optimization of a class of relational expressions", ACM Transactions on Database Systems (TODS), Volume 4 , Issue 4 , Pages: 435 - 454, December 1979.

[2] Atkinson *et al*, (2003), "Database Access and Integration", chapter commissioned for the second edition of The Grid, Foster & Kesselman. 1st April 2003. http://www.ogsadai.org.uk/documentation/publications. [Accessed, October 17th 2006]

[3] Atkinson *et al*, (2003), "Grid Database Access and Integration: Requirements and Functionalities", Global Grid Forum (2003) http://www.gridforum.org/documents/GFD.13.pdf., [Accessed, September 15th ,2005]

[4] Atkinson, (2004), "Data Access and Integration", European Research Consortium for Informatics and Mathematics, Number 59, October 2004

[5] Baru *et al*, (1998), "The SDSC Storage Resource Broker", Proc. Of CASCON'98 Conference, Toronto, Canada (1998)

[6] Berners-Lee *et al*, (2001), "The Semantic Web". Scientific American Magazine. Retrieved on 26 March 2008.

[7] Bunge, (1977), "Ontology I: The Furniture of the World", Treatise on Basic Philosophy, Vol.3, D.Reidel Publishing Co., Inc., New York, 1997.

[8] Cali *et al*, (2002), "Data Integration under Integrity Constraints", Proceedings of the 14th Conference on Advanced Information Systems Engineering (CAiSE 2002), 2002.

[9] Chandrasekaran *et al*, (1999), "What Are Ontologies, and Why Do We Need Them?", Intelligent Systems and Their Applications, IEEE, vol. 14, no. 1, 1999, pp. 20-26.

[10] Castellanos *et al* (1991), "Suitability of data models as canonical models for federated databases". ACM SIGMOD Record, 20(4):44 - 48, 1991.

[11] Chervenak *et al*, (2000), "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets", Journal of Network and Computer Applications, vol. 23, no. 3, pp. 187–200, 2000.

[12] Cutting *et al*, (1993), "Scatter/gather: A cluster-based approach to browsing large document collections." In 16th International ACM SIGIR Conference on Research and Development in IR, pp 126-135, 1993.

[13] Fensel *et al*, (1998), "Ontobroker: How to make the WWW Intelligent", Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management (KAW'98, Banff, Canada, Apr.).

[14] Foster *et al*, (2001), "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", Int. J. Supercomp. App., 15(3):200–222, 2001.

[15] Foster *et al*, (2002), "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Open Grid Service Infrastructure WG, GGF 2002.

[16] Fox *et al*, (2003), "e-Science gap analysis". technical report. Available as http://grids.ucs.indiana.edu/ptliupages/publications/GapAnalysis30June03v2.pdf.

[17] Gruber, (1993), "A Translation Approach to Portable Ontologies", *Knowledge Acquisition*, pp. 199-220.

[18] Guarino , (1998), "Formal Ontology in Information Systems",Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15.

[19] Hakimpour *et al*, (2002), "Global schema generation using formal ontologies", In Proceeding ER02, volume 2503 of LNCS, pages 307–321. Springer-Verlag, 2002.

[20] Koutrika, G. (2005). Heterogeneity in digital libraries: Two sides of the same coin. DELOS Newsletter. http://delos-old.isti.cnr.it/newsletter/issue3/feature2/.

[21] Lenzerini (2002), "Data Integration: A Theoretical Perspective", In Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002), 233-246. New York: Association for Computing Machinery. Madison, Wisconsin

[22] Pearson, (2002) , "Data Requirements for The Grid: Scoping Study Report", UK DBTF working paper,. February 2002 (presented at GGF4).

[23] Quix, (2004), "Quality-oriented and Metadata-driven Integration in Information Grids", 2nd

IST Workshop on Metadata Management in Grid and P2P Systems (MMGPS) - Models, Services, Architectures, London, UK, Dec. 2004

[24] Ram *et al*, (2002). "CREAM: A Mediator Based Environment for Modeling and Accessing Distributed Information on the Web". In Proceedings of the 19th British National Conference on Databases: Advances in Databases (July 17 - 19, 2002). B. Eaglestone, S. North, and A. Poulovassilis, Eds. Springer-Verlag, London, 58-61.

[25] Roure (2003), "The Semantic Grid: A Future e-Science Infrastructure," Grid Computing: Making the Global Infrastructure a Reality, F. Berman, G. Fox, and T. Hey, eds., John Wiley & Sons, 2003

[26] Salton *et al*, (1983), "Introduction to Modern Information Retrieval", Mc Graw-Hill Computer Science Series, 1983

[27] Sagiv *et al*, (1980). "Equivalences among relational expressions with the union and difference operators". Journal of the ACM, 27(4):633-655, October 1980

[28] Sheth *et al*, (1990), "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys, Vol. 22, No. 3, September 1990

[29] Ullman (1997), "Information Integration Using Logical Views". Proceddings of 6th International Conference, Delphi, Greece, 1997.

[30] Watson, (2001), "Database and The Grid", Technical Report CS-TR-755, University of Newcastle, 2001.

[31] Wells, (2005), "Grid Database Design", Auerbach Publications Taylor & Francis Group, 2005.

[32] Wiederhold, Gio (1992), "Mediators in the Architecture of Future Information Systems". IEEE Computer, 25 (3). pp. 38-49.